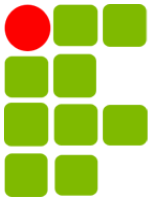


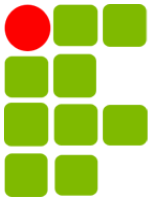
Instituto Federal de Educação, Ciência e Tecnologia da Paraíba
Campus Campina Grande
Coordenação do Curso Superior de Tecnologia em Telemática

# Monitoramento e Controle: Sistema de Bombeamento Hidraúlico

Luciano Estrela de Lacerda

Orientador: Fagner de Araujo Pereira

Campina Grande, dezembro de 2023



Instituto Federal de Educação, Ciência e Tecnologia da Paraíba
Campus Campina Grande
Coordenação do Curso Superior de Tecnologia em Telemática

# Monitoramento e Controle: Sistema de Bombeamento Hidraúlico

Luciano Estrela de Lacerda

Monografia apresentada à Coordenação do Curso de Telemática do IFPB - Campus Campina Grande, como requisito parcial para conclusão do curso de Tecnologia em Telemática.

Orientador: Fagner de Araujo Pereira

Campina Grande, dezembro de 2023

L131m Lacerda, Luciano Estrela de
Monitoramento e controle: sistema de bombeamento hidráulico / Luciano Estrela de Lacerda. - Campina Grande, 2023.
35f. : il.

Trabalho de Conclusão de Curso (Curso Superior de Tecnologia em Telemática) - Instituto Federal da Paraíba, 2023.
Orientador: Prof. Fagner de Araujo Pereira

1. Internet das coisas - monitoramento de motobomba hidráulica 2. Software livre - ScadaBR 3. Arduino I. Pereira, Fagner de Araujo II. Título.

CDU 004.738

# Monitoramento e Controle: Sistema de Bombeamento Hidraúlico

**LUCIANO ESTRELA DE LACERDA**

Fagner de Araujo Pereira

---

Orientador

José Antônio Cândido Borges da Silva, D.Sc.

---

Membro da Banca

Katyusco de Farias Santos

---

Membro da Banca

Campina Grande, Paraíba, Brasil
Dezembro/2023

*Dedico este trabalho primeiro a Deus e depois a mim mesmo, por nunca desistir.*

*"Após vários dias de chuvas e nublados, é certo que o sol irá brilhar novamente."*
*Luciano Estrela*

# Agradecimentos

Agradecer aos docentes do curso de Telemática e toda a equipe que compõe a instituição IFPB Campus Campina Grande. Em especial me dirijo aos docentes, que ao longo de todos esses anos de graduação passaram não somente conhecimento mas também a experiência profissional, a ética educacional e o respeito aos valores sociais no ambiente acadêmico.

# Resumo

Sistemas de monitoramento de ativos tornaram-se realidade dentro da indústria de manufatura. É inconcebível para qualquer segmento industrial, com foco na redução de custos, sem que haja monitoramento e gestão de seus ativos. A internet das coisas (IoT) vem proporcionando a inclusão de ativos de modo online com custos cada vez mais reduzidos e essa tendência fica clara porque hoje várias empresas vendem esse tipo de serviço. Cada vez mais, serviços online são buscados para gestão de ativos até mesmo para clientes não industrias.

Este trabalho documenta minha experiência no desenvolvimento de um sistema computacional de supervisão para o monitoramento e controle de um conjunto motobomba instalado no Condomínio Residencial Sierra, situado em Campina Grande. O projeto teve como parceiro a companhia de água e esgotos da paraíba (Cagepa). Nesse sistema, doravante chamado de supervisório, são obtidas informações por meio de um servidor (Arduino) sobre a situação da bomba (acionada, parada ou em falha) e permite o controle remoto de acionamento da bomba. Além disso, o servidor também obtém informações de um transmissor de pressão instalado após a bomba (jusante). Dessa forma, é fornecido ao controlador do sistema (aquele que monitora) uma interface simples mas que permite, em tempo real, verificar o funcionamento e situação de abastecimento do condomínio, razão principal desse projeto.

Nesse trabalho foram desenvolvidos dois *design* de supervisórios simultâneos mas com as mesmas funções: o primeiro foi desenvolvido com o software ScadaBR e ficou disponível ao condomínio. A escolha do ScadaBR foi feita por ser um software livre, gratuito e de código-fonte aberto. O ScadaBR é uma ótima escolha para supervisórios, aplicações de automação, aquisição de dados e controle onde o custo dessa ferramenta seja impactante no orçamento do projeto. O segundo foi uma tela de supervisão feita no supervisório WinCC utilizado pelo Centro de Controle Operacional (CCO) da Cagepa em Campina Grande.

**Palavras-chave:** Supervisório, ScadaBR, Arduino.

# Abstract

Asset monitoring systems have become a reality within the manufacturing industry. It is inconceivable for any industrial segment, focused on reducing costs, without monitoring and managing its assets. The internet of things (IoT) has been enabling the inclusion of assets online at increasingly reduced costs and this trend is clear because today several companies sell this type of service. Increasingly, online services are sought after for asset management even for non-industrial clients.

In this work is presente the development a computer supervision system for monitoring and controlling a motor pump set installed at Condomínio Residencial Sierra, located in Campina Grande. The project had as a partner the water and sewage company of Paraíba (Cagepa). In this system, hereinafter called supervisory, information is obtained through a server (Arduino) about the pump's status (activated, stopped or failing) and allows remote control of the pump's activation. Additionally, the server also obtains information from a pressure transmitter installed after the pump (downstream). In this way, the system controller (the one who monitors) is provided with a simple interface that allows, in real time, to check the operation and supply status of the condominium, the main reason for this project.

In this work, two simultaneous supervisory projects were developed but with the same functions: the first was developed with the ScadaBR software and was available to the condominium. ScadaBR was chosen because it is free, open-source software. ScadaBR is a great choice for supervisory, automation, data acquisition and control applications where the cost of this tool impacts the project budget. The second was a supervision screen made on the Scada WinCC supervisory used by the Cagepa Operational Control Center (OCC) in Campina Grande.

**Keywords:** Supervisory, ScadaBR, Arduino.

# Sumário

# Lista de Abreviaturas

| | |
|---|---|
| Cagepa | *Companhia de Água e Esgotos da Paraíba* |
| CCO | *Centro de Controle Operacional* |
| CLP | *Controlador Lógico Programável* |
| EoIP | *Ethernet Over IP* |
| HMI | *Human Machine Interface* |
| IEEE | *Instituto de Engenheiros Eletricistas e Eletrônicos* |
| IoT | *Internet Of Things* |
| LAN | *Local Area Network* |
| PCI | *Placa de Circuito Impresso* |
| SCADA | *Supervisory Control and Data Acquisition* |
| SQL | *Structured Query Language* |
| UTR | *Unidade de Transmissão Remota* |
| VPN | *Virtual Private Network* |
| WLAN | *Wireless Local Area Network* |

# Lista de Figuras

# Lista de Tabelas

# Capítulo 1

# Introdução

Atualmente conceitos como eficiência energética, otimização de processos e produtividade motivam e direcionam setores de produção e também de serviços.

Na prática, a sofisticação de processos ou serviços é a ideia chave de uma regra de negócio e sempre, com ela, temos redução de custos, aumento de lucros e possivelmente agrega-se algum tipo de "humanização do trabalho". Alguns benefícios são redução de trabalho repetitivo e menos exposição a situações perigosas, por exemplo.

[Silveira e Lima 2003] dissertam que todos esses conceitos acima conduzem a automação como método base de criar mecanismos que produzem o melhor serviço ou produto de modo a alcançar também o menor custo. É necessário, portanto, que os princípios de automação busquem:

- Melhorar a produtividade;
- Melhorar as condições de trabalho, eliminando situações perigosas;
- Realizar operações de difícil controle intelectual ou manual;
- Simplificar a operação de sistemas, facilitando a operação sobre o processo.

Ainda segundo [Silveira e Lima 2003] uma solução de automação consiste de duas partes sendo uma operacional e outra de controle. A operacional representa os ativos do sistema que atuam sobre o processo. Exemplos de elementos que compõe esta parte são dispositivos de acionamento e pré-acionamento como: motores, cilindros, bombas, válvulas e pistões. Aqui também estão inclusos dispositivos como sensores e transmissores que representam as variáveis do processo.

A parte de controle representa o software do sistema geralmente implementado em um Controlador Lógico Programável (CLP) ou numa plataforma microcontrolada que se preste a esta função. Um fato de destaque é que antes do surgimento da microeletrônica, o software era realizado com chaves eletromecânicas (relés) e temporizadores. Ao longo do avanço tecnológico, principalmente da microeletrônica, os computadores foram incluídos de maneira natural na realização desse tipo de controle, principalmente pelo poder computacional e o custo gradativamente reduzido.

Uma possível conclusão nesse ponto é que num projeto de automação é necessário fornecer ao processo, elementos que sejam capazes de executar um algoritmo (software), elementos que

obtenham informações específicas dentro do processo (entradas) e atuadores que promovam modificações sobre grandezas mensuradas anteriormente (saídas). Para implementar um software de automação são utilizados CLPs, hardwares específicos de aplicação ou mesmo uma plataforma microcontrolada de uso mais geral a depender do caso.

As informações do processo são obtidas por meio de sensores ou transmissores. Estes dispositivos desempenham a função de converter uma grandeza física em elétrica, a qual será recebida pelo elemento que executa o algoritmo. Para modificar o processo é necessário a utilização de atuadores que são dispositivos que convertem sinais elétricos em grandezas físicas. Dentre estas são comumente utilizados: o calor, força e torque.

Atualmente estamos na "Indústria 4.0". A principal mudança dessa nova fase é que a gestão operacional de processos complexos também são viabilizadas pelo monitoramento das variáveis do processo. Informações em tempo real acerca de grandezas físicas, do *status* de atuadores e o controle por meio de comandos ou parâmetros permitem estabelecer uma relação operacional eficiente num processo, principalmente quando os cenários da planta para uma solução de automação definitiva são complexos. O monitoramento do processo pode também se valer de alarmes e eventos acionados pela extrapolação de um limite estabelecido para uma variável de processo ou numa situação prevista que requer atendimento priorizado. Nesse modelo, o controlador do sistema pode intervir numa planta complexa, por meio do software, através de parâmetros ou comandos em regime integral de supervisão (chamado de 24/7) ou sob demanda por meio de gráficos.

Sistemas SCADA (Supervisory Control and Data Acquisition) englobam um conjunto de tecnologias (equipamentos, softwares e padrões) especialmente desenvolvidas para monitorar e controlar processos industriais. O conceito de SCADA pode ser estendido a outras áreas como laboratórios, tráfego e automação predial/residencial. O principal objetivo é propiciar uma interface de alto nível para um operador, que o informe sobre variáveis do processo e eventos de importância.

Conforme [Vianna, BRINGHENTI e MARTINS 2008], um sistema SCADA típico oferece uma série de vantagens que podem ser destacadas:

- Comunicação com equipamentos em diferentes protocolos;
- Registro e Relatórios (data logger);
- Alarmes e Eventos;
- Interface Gráfica para Operação de Processos (HMI ou Interface Homem-Máquina);
- Integração com softwares externos.

Por estes poucos e muitos outros motivos, sistemas SCADA tem sido procurado por empreendedores na busca por inovar, atualizar suas indústrias ou processos criando assim um mercado a ser desenvolvido e explorado por tecnólogos e engenheiros de todos país.

Uma das atividades que acompanham este paradigma é o setor de saneamento no qual tem por base os reservatórios, as redes de distribuição e as instalações elevatórias formadas principalmente pelos conjunto motobombas [GOMES 2009].

A planta em que o sistema SCADA será desenvolvido para controle e monitoramento é um conjunto motobomba formada por um motor trifásico, uma bomba centrífuga, inversor de frequência, quadro elétrico e cabos elétricos diversos. As informações da mecânica do sistema são pouco relevantes para o projeto de monitoramento, contudo, complementam este trabalho.
Conjunto motobomba formado por:

- Bomba centrífuga ksb multiestágio ($Q = 18\ m^3h, H = 70\ mca$);
- Motor de indução trifásico 380 V, 2 pólos (3510 rpm) e 5 CV;
- Inversor WEG CFW300 3.7 kW.

O objetivo desse conjunto motobomba é melhorar o abastecimento de água do condomínio. Para isso, o sistema motobomba foi instalado junto ao hidrômetro (na entrada de água do condomínio) e atua fornecendo aumento de energia hidraúlica (pressão/vazão) a jusante da bomba em horários específicos de funcionamento. Os termos jusante e montante em sistemas de bombeamento são usados para se referir ao que está antes da bomba (montante) e após a bomba (jusante). Por conseguinte, o tempo de recarga de todo o condomínio, cujo os imovéis possuem cisternas, fica consideravelmente menor. O conjunto motobomba é acionado por um inversor de frequência que, nesse caso, é utilizado apenas como elemento de comando para acionar o motor. Em parceria com a Cagepa (Companhia de Água e Esgotos da Paraíba), financiadora de todo o projeto, o monitoramento de tal sistema motobomba deve também ser integrado ao atual sistema SCADA utilizado pelo Centro de Controle Operacional (CCO) já existente na companhia. Sistema esse que é todo baseado na plataforma Siemens (CLPs e Supervisório).

**Figura 1.1:** *Conjunto motobomba Condomínio Residencial Sierra*

*Fonte: Autor*

O Centro de Controle Operacional da Cagepa em Campina Grande é formado por um servidor de aplicação SCADA e dois clientes (WinCC-Siemens). A infraestrura de comunicação compreende uma rede TCP/IP do tipo bridge ethernet LAN/WLAN (IEEE 802.3 e 802.11) e duas VPNs do tipo EoIP. De maneira muito conveniente, os próprios reservatórios elevados da companhia na cidade foram utilizados como "torres"para instalações dos rádios de *Access Points* sem fio para comunicação entre as unidades operacionais (*sites*). Dessa forma, o sinal de rádio viaja entre os links situados sobre os reservatórios e alcança a rede principal que fica situada no CCO. Por fim, temos as unidades remotas chamadas de UTRs. As UTRs são armários elétricos dotados de CLP-Siemens e sensores utilizados no controle e monitoramento da unidade operacional, que podem ser os reservatórios ou instalações elevatórias. Em última análise, as UTRs são servidores que trocam dados com os clientes SCADA no CCO acerca dos níveis, pressão, vazão e permitem ao controlador do CCO o acionamento de motores e fechamento de válvulas.

O CCO é uma visão de negócio mais recente da companhia visando a gestão operacional simultânea de várias unidades na cidade a partir de um ou dois controladores e um adendo a automação em virtude de problemas com o modelo anterior baseado em operadores. A fim de melhorar a qualidade do serviço e o tempo de resposta ao cliente, o controlador do sistema tem a sua disposição informações em tempo real da planta de abastecimento e pode rapidamente atuar sobre a mesma. Tecnicamente falando a Cagepa é a maior indústria em operação na cidade onde sua planta de instalação tem os limites do município. O produto é água tratada e a logística é feita pela rede de distribuição. Ter informações específicas sobre a situação em vários pontos de abastecimento e poder atuar de imediato sobre esses pontos só é possível através da tecnologia de redes.

## 1.1 Justificativa e Relevância do Trabalho

Esse trabalho é uma síntese das competências adquiridas ao longo de minha graduação em diversas disciplinas. Consolida meu amadurecimento profissional no uso de técnicas de redes e programação. Para o Curso de Telemática, este trabalho se identifica como mais um ramo de aplicação dentre muitos. Por fim, esse trabalho é a base para futuros projetos em temas afins.

## 1.2 Objetivos

### 1.2.1 Objetivo Geral

Nessa parte são descritos os objetivos do trabalho de maneira a esclarecer as motivações que levaram a escolha dos métodos em busca dos resultados.

Este trabalho de conclusão de curso visa utilizar os conhecimentos aprendidos durante minha formação acadêmica, em especial nas disciplinas de Programação 3 (professor Victor), Microcontroladores (professor Fagner) e Comunicações Sem Fio (professor Jeronimo) para projetar um servidor remoto simples e programar um sistema de supervisão e aquisição de dados para supervisionar o funcionamento de um conjunto motobomba instalado no Condomínio Residencial Sierra localizado em Campina Grande. Esse projeto será integrado ao sistema de bombeamento e adiciona um recurso de software que possibilita a visualização de diversas informações e controle mais ágil sobre o processo.

### 1.2.2 Objetivos Específicos

Nesse ponto é definido os objetivo específicos do projeto:

- Pesquisar sobre tecnologias para o desenvolvimento de sistemas do tipo SCADA;
- Desenvolver uma HMI que seja perceptível as vantagens para monitoramento remoto de processos;
- Estudar o protocolo ModBus TCP/IP para comunicação entre sistemas distintos;
- Projetar um *hardware* simples que compõe o servidor Arduino;
- Programar um servidor Arduino para comunicação permanente e confiável com outro sistema;
- Calcular e implementar um rádio enlace curto para comunicação de dados com o servidor Arduino.

## 1.3 Organização do Documento

A fim de melhor organizar o trabalho, foi feito a seguinte divisão:

- Capítulo 2: Fundamentação Teórica. Nesse capítulo são postos os conceitos mais relevantes para o entendimento do trabalho.
- Capítulo 3: Desenvolvimento/Resultados: Nesse capítulo são mostrados e apresentados os resultados do trabalho.
- Capítulo 4: Conclusão: Nesse ponto são apresentadas as conclusões do trabalho.
- As referências e os anexos estão no final do documento.

# Capítulo 2

# Fundamentação Teórica

Neste capítulo são apresentados os conceitos e fundamentos que embasaram o desenvolvimento do sistema proposto. Em especial é exposto o funcionamento e o código fonte base do servidor Arduino, do protocolo de comunicação Modbus TCP/IP e sobre os conceitos relacionados com sistemas do tipo SCADA. Propositalmente eu não incluo a fundamentação teórica sobre o funcionamento do conjunto motobomba por ser necessário conhecimento específico de mecânica dos fluidos.

## 2.1 Servidor Arduino

Ao cursar a disciplina microcontroladores com o professor Fagner A. Pereira, foi possível adquirir competências relacionadas ao funcionamento, arquitetura e programação nesses dispositivos. Na disciplina foram abordados os conceitos relacionados ao estudo de microprocessadores e microcontroladores, funcionamento da CPU, set de instruções, estudos dos registradores, técnicas de programação etc. Esta disciplina forneceu a base para testar pequenos projetos utilizando fabricantes e arquiteturas de micros diferentes. Nesse trabalho foi utilizado o micro da arquitetura AVR do fabricante ATMEL. Especificamente foi usado o micro ATMEGA328P que integra a plataforma Arduino Uno. Os principais motivos que levaram a escolha da plataforma e do micro foram:

- Arquitetura RISC e desempenho eficiente para o projeto;
- Entradas analógicas e I/O suficientes;
- *Ethernet Shield* com interface padrão IEEE 802.3;
- *Shield* de Bornes para montagem mais profissional;
- API de *sockets*.

Como boa prática de projeto, a escolha da plataforma Arduino Uno deve ser sempre avaliada em conjunto com os recursos disponíveis do micro ATMEGA328P na proposta de solução.

**Figura 2.1:** *Plataforma Arduino e micro ATMEGA328P*


*Fonte: Autor*

A ideia do servidor Arduino surgiu naturalmente durante a disciplina de Programação 3 com os conceitos de *sockets* e arquitetura cliente-servidor (UDP/TCP), juntamente com as diversas abordagens de programação feitas pelo professor Victor A. P. Oliveira. Na ocasião da disciplina foi possível implementar *sockets* no Arduino a partir de um recurso de *hardware* conhecido como *Ethernet Shield*. Aqui cabe a ressalva de que *shields* para o Arduino são placas de expansão que podem ser conectadas na parte superior dele e que lhe adicionam novas funcionalidades.

O *Ethernet Shield* é baseado no chip Wiznet W5100. O W5100 implementa a pilha de protocolos TCP/IP tanto para o transporte com TCP ou UDP. Ele possui 16kB de memória para *buffer* dos dados provinientes das comunicações e suporta até 4 conexões distintas simultâneas. Dessa forma, o servidor Arduino desse trabalho pode atender no máximo 4 clientes simultâneos, sendo a conexão do quinto rejeitada.

**Figura 2.2:** *Arduino e Ethernet Shield empilhados*

*Fonte: Autor*

O conceito chave do funcionamento desse trabalho é, sem dúvida, o de *sockets*. O *sockets* é a abstração por meio da qual processos em máquinas distintas podem se comunicar através de uma rede. De maneira conceitual, podemos dizer que um processo fica ouvindo as solicitações (*Server*) e outro que envia as requisições (*Client*). Como *socket* é independente do protocolo usado na comunicação, podemos livremente escolher o protocolo ou, até mesmo, criar nosso próprio protocolo.

A criação de um *socket* de escuta, ou servidor, no Arduino é feito pela instânciação de um

objeto da classe *EthernetServer*, por meio do include da biblioteca <*Ethernet.h*>. O trecho de código a seguir exemplifica como colocar o Arduino em modo de escuta (servidor) para receber as solitações de rede:

```
#include <SPI.h>
#include <Ethernet.h>

#define PORT 5000 // 0-65535

byte mac[] = { 0xAA, 0xBB, 0xCC, 0xDD, 0xEE, 0x01 };

IPAddress ip(192,168,0,140);
IPAddress gateway(192,168,0,1);
IPAddress subnet(255,255,255,0);

EthernetServer server(PORT);
EthernetClient client;

void setup() {
  Ethernet.begin(mac, ip, gateway, subnet);
}

void loop() {
  client = server.available(); // retorna a qtd de bytes lidos do buffer.
   client = 0 => nenhum cliente.

  if (client) {
    if ( client.available() ) { // se houver bytes para serem lidos, entao ...
      /* tratar os bytes lidos aqui */
      ;
    }
    else { // se todos os bytes foram lidos, entao ...
      client.stop(); // o fim da conexao sempre inicia do lado cliente
    }
  }
}
```

**Listing 2.1:** *Base Code Server*

O código fonte 2.1 em *C++* foi complementado para objetivo desse projeto. No mínimo, seis versões foram feitas para resolver *bugs* e condições excepcionais do funcionamento. Essa tarefa de testar a qualidade do funcionamento de um produto para determinada tarefa é conhecido como comissionamento. O produto que passa em tal etapa é dito como “comissionado”.

## 2.2 Modbus TCP/IP

O protocolo Modbus TCP/IP (ou apenas Modbus IP) é uma variante da família Modbus de protocolos de comunicação simples e neutros em termos de fornecedor destinado à supervisão e controle de equipamentos de automação. É um protocolo versátil e amplamente usado para interoperabilidade de equipamentos de automação em geral. O Modbus IP troca mensagens por meio de pacotes em um ambiente de 'Intranet' ou 'Internet' usando a pilha TCP/IP. O uso mais comum desse protocolo é para estabelecer comunicação ethernet de PLCs e supervisórios de fabricantes distintos [Swales 1999]. A escolha desse protocolo para comunicação nesse trabalho foi devido a essas características e por ser amplamente usado pelos fabricantes na indústria.

No Modbus IP uma única conexão pode realizar quantas requisições independentes se queira. É possível conexões simultâneas e cabe ao cliente escolher entre reconectar conforme necessário ou reutilizar uma conexão já existente. Um questionamento interessante é por qual razão o Modbus IP usa o protocolo de transporte TCP (orientado a conexão) ao invés de UDP (orientado a datagramas)? O Modbus com UDP existe mas é menos confiável que o Modbus TCP porque não garante a entrega ou a sequência correta dos pacotes. Contudo é mais rápido e requer menos largura de banda. Com TCP também é possível melhorar o desempenho as mudanças na rede e permitir que recursos de segurança, como *Firewalls* e *Proxies*, sejam facilmente adicionados.

### 2.2.1 Codificação dos dados

O Modbus usa uma representação *big-endian* para os *bytes* nos pacotes de endereços ou dados. Isto significa que quando uma quantidade numérica maior que um único *byte* é transmitida, o *byte* mais significativo é enviado primeiro. Na tabela 2.1, o *byte* $B_1$ é transmitido primeiro.

**Tabela 2.1:** *Formato de transmissão dos bytes*

| Lenght | Data | $B_1$ | $B_2$ | $B_3$ | $B_4$ |
|---|---|---|---|---|---|
| 16-bits | 0x1234 | 0x12 | 0x34 | | |
| 32-bits | 0x12345678L | 0x12 | 0x34 | 0x56 | 0x78 |

*Fonte: Autor*

### 2.2.2 Estrutura do protocolo

Esta seção descreve a ordem dos *bytes* que contém uma solicitação ou resposta do Modbus IP. O entendimento dessa estrutura é fundamental para o desenvolvimento do código-fonte do servidor Arduino. Após o recebimento de uma solicitação, cada *byte* lido pelo *socket* será intepretado e contém uma função específica dentro do protocolo. É importante observar que a estrutura dos *bytes* de solicitação e resposta, do código de função até o final da parte de dados, têm exatamente o mesmo layout e significado que tem as outras variantes Modbus, como

- Modbus serial port - ASCII encoding;

- Modbus serial port - RTU (binary) encoding.

As únicas diferenças entres esses casos são a forma de verificação de erros e interpretação de endereço. Com o Modbus IP todas as solicitações são enviadas via TCP na porta 502 por padrão. As solicitações normalmente são enviadas de maneira *half-duplex*, isto é, uma solictação do cliente seguido de uma resposta do servidor para cada conexão. Não há benefício em enviar solicitações adicionais em uma única conexão enquanto uma resposta estiver pendente. Uma estratégia possivel para obter altas taxas de transferência é estabelecer múltiplas conexões TCP para o mesmo alvo [Swales 1999].

O campo 'endereço escravo' de versões anteriores foi substituído por um único *byte* 'Unit Identifier' que pode ser usado para comunicar através de dispositivos como pontes e gateways que usam um único endereço IP para suportar vários unidades finais independentes. O Modbus em versões anteriores, baseadas em transmissões seriais, usava a terminologia *Master* para cliente e *Slave* para servidor.

A solicitação e resposta são prefixadas por seis *bytes* como segue:
*byte* 0: identificador da transação – copiado pelo servidor – geralmente 0
*byte* 1: identificador da transação – copiado pelo servidor – geralmente 0
*byte* 2: identificador de protocolo = 0
*byte* 3: identificador de protocolo = 0
*byte* 4: campo de comprimento (*byte* superior) = 0 (já que todas as mensagens são menores que 256)
*byte* 5: campo de comprimento (*byte* inferior) = número de *bytes* seguintes
*byte* 6: identificador da unidade*UI* (anteriormente 'endereço escravo')
*byte* 7: código de função Modbus
*byte* 8 ativado: dados conforme necessário

Por exemplo, uma solicitação do cliente ao servidor poderia ser ler 1 registro (2 *bytes*) com *off-set* de 4 apartir do endereço base (0x00) com UI igual a 9. Supondo que aquele registro tivesse o valor 5 a resposta do servidor seria:

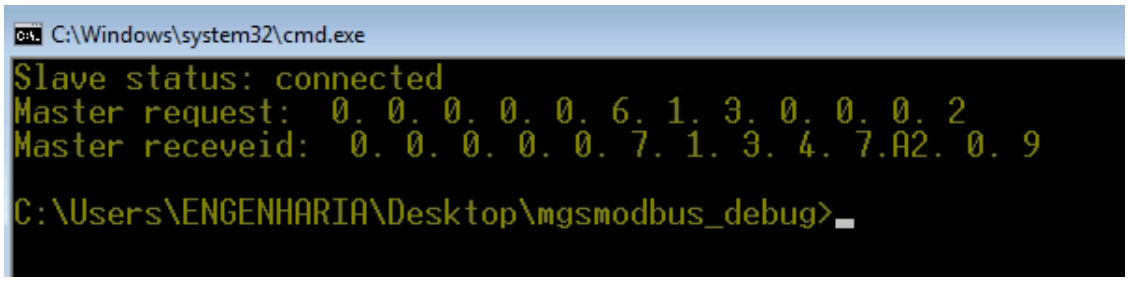
solicitação: 00 00 00 00 00 06 09 03 00 04 00 01
resposta: 00 00 00 00 00 05 09 03 02 00 05

Dentre as diversas funções do Modbus, valor do *byte* 7, e para os propósitos desse trabalho, vamos nos limitar as duas funções seguintes:

- Read multiple registers (FC 3);
- Write single register (FC 6).

Durante a fase de estudo do protocolo Modbus foi necessário criar uma maneira de visualizar a troca correta de mensagens entre cliente e servidor para propósitos de *debug* do código-fonte do

**Figura 2.3:** *Troca de mensagens entre cliente Windows e servidor Arduino*




*Fonte: Autor*

servidor Arduino. Na figura 2.3 a seguir podemos ver a troca dos *bytes* entre um cliente *Windows* e o servidor Arduino. Cada *byte* é separado por um ponto e o valor de cada *byte* está em hexadecimal.

Nessa solicitação feito pelo cliente *Windows* podemos ver os primeiros seis *bytes* que prefixam tanto requisição como resposta. O valor do *byte* 5, a contagem inicia em 0, contém a quantidade de *bytes* seguintes a partir dele. O *byte* 6 informa o *UI* do servidor Arduino. O *byte* 7 contém o código da função Modbus que nesse caso é 3, ou seja, ler múltiplos registros. A partir do *byte* 8 especificamos na solitação o *off-set* e a quantidade de registros que desejamos ler. Na resposta, o valor do *byte* 8 contém o comprimento em *bytes* da solicitação que deve ser o dobro da quantidade de registros (cada registro possui dois *bytes*).

## 2.3 ScadaBR

ScadaBR é um software livre, gratuito e de código-fonte aberto, para desenvolvimento de aplicações de automação, aquisição de dados e controle supervisório. O alicerce do software nasceu em 2005 a partir do trabalho de conclusão de curso do graduando Victor Rocha Push, no curso Engenharia de Controle e Automação (UFSC). Em 2006 a MCA Sistemas (atualmente sensorweb) inicia o desenvolvimento do ScadaBR, visando preencher uma lacuna de mercado percebida - a não existência de softwares livres profissionais para aquisição de dados e controle supervisório. Ao longo dos anos o projeto foi sendo melhorado pela MCA até que em 2009 ocorre o lançamento oficial da primeira versão, juntamente com a página *www.scadabr.org.br*. Atualmente o software está na versão 1.2 que, inclusive, foi utilizada nesse trabalho.

De maneira resumida, sistemas SCADA (*Supervisory Control and Data Acquisition*) englobam um conjunto de tecnologias (equipamentos, softwares e padrões) especialmente desenvolvidas para monitorar e controlar processos Industriais. Entretanto, o conceito de SCADA não fica somente associado às industrias. Outras áreas como laboratórios, tráfego, automação predial, semafórica e outras podem ser benefiadas com o uso de sistemas SCADA.

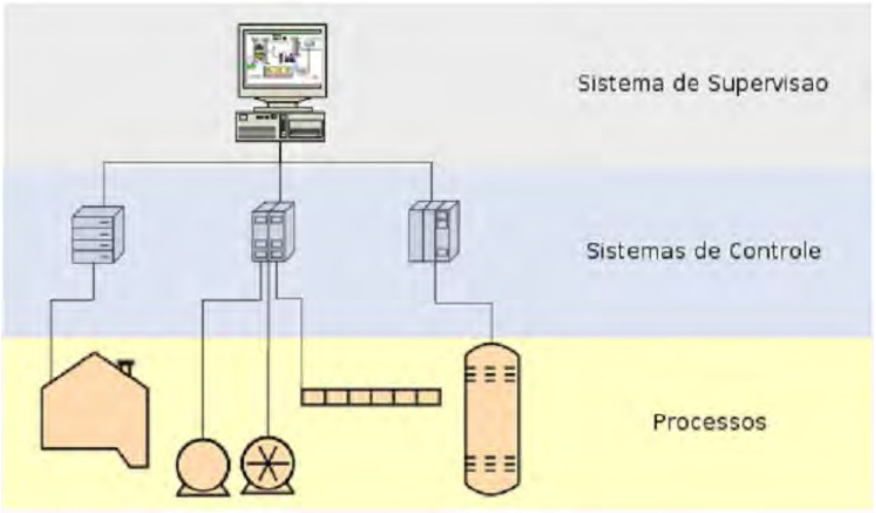
ScadaBR segue o mesmo padrão de sistemas SCADAs típicos. A arquiteruta mais simples, figura 2.5, consiste em uma estrutura de 3 camadas. Numa visão *top-down*, a primeira camada é composta de um sistema computacional supervisório. Nesse estão todas as partes de alto nível do sistema incluindo o banco de dados, execução de *scripts*, de onde parte as requisições/solicitacões,

**Figura 2.4:** *SCADA - Aplicações comuns*


*Fonte: Autor*

os sinóptico da planta, protocolos de comunicação somente pra citar alguns. A camada intermediária comprende os sistemas de controle. Esses, por sua vez, são compostos pelos CLPs ou qualquer outro *hardware* designado para esse fim. O objetivo dessa camada é o controle e leitura, objetos de campo e sensores respectivamente. A última camada que se segue são os objetos do processo. Sensores e atuadores de vários tipos e modelos são típicos dessa camada.

**Figura 2.5:** *SCADA - Camadas*




*Fonte: Autor*

# Capítulo 3

# Desenvolvimento da Solução

Nesta seção serão apresentadas as etapas para o desenvolvimento dos sistemas SCADA.

## 3.1 Simatic WinCC e ScadaBR

Simatic WinCC ou somente WinCC e ScadaBR são ambos softwares do tipo SCADA. Esses softwares possuem uma interface de operação baseada em PC onde é possível programar um sistema de monitoramento para visualização e controle de processos, fluxos de produção, máquinas ou mesmo toda a planta de operação. Ambos WinCC e ScadaBR compartilham de ferramentas comuns de sistemas SCADA típicos que já foram listadas.

WinCC é um software de engenharia proprietário para configurar painéis SIMATIC, PCs industriais SIMATIC e PCs padrão com o software de visualização WinCC Runtime Advanced. O termo SIMATIC compreende uma série de controladores lógicos programáveis e sistemas de automação, desenvolvidos pela Siemens. Introduzida em 1958, a série passou por quatro gerações principais sendo a última a geração SIMATIC S7. A série é destinada à automação e produção industrial. O portfólio da Siemens inclui soluções para qualquer tipo de indústria de manufatura e se consagra como marca no topo no segmento de automação industrial.

**Figura 3.1:** *WinCC overview*


*Fonte: Autor*

ScadaBR é um software que possui a maiorias das ferramentas que já existem em softwares comerciais SCADA, como o WinCC listado acima. O ScadaBR pretende oferecer todas as funcionalidades de um sistema SCADA tradicional. Este tipo de software(SCADA) existe desde o final dos anos 60 e é a peça fundamental em qualquer tipo de aplicação computadorizada que envolva máquinas, controladores programáveis (CLP's), acionamentos eletrônicos e sensores.

**Figura 3.2:** *ScadaBr overview*


*Fonte: Autor*

Em consulta as informações disponíveis pela Siemens e do projeto ScadaBR, ambos os softwares proporcionam benefícios e possuem características especificadas abaixo.
Benefícios:

- Conexão com a maioria dos equipamentos (PLC's, remotas, concentradores de dados) de mercado;
- Redução no tempo de desenvolvimento e manutenção, através da padronização das aplicações com o uso de bibliotecas*;
- Retorno rápido e duradouro do investimento.

Principais Características:

- Multiusuários e multiprojetos: Permite editar e executar diversos projetos simultaneamente*;
- Bibliotecas de objetos gráficos e estruturas de dados reutilizáveis;
- Redundância nativa com sincronismo de dados históricos e alarmes*;
- Estação de engenharia para alterações in *RunTime**;

- Editor de telas completo e poderoso*;
- Segurança e compactação na transmissão de dados;
- Flexibilidade na gestão de alarmes e eventos;
- Ferramenta de scripts;
- Acesso a bancos de dados como MySQL;
- Ferramenta de logs, consultas e relatórios integrada.

obs.: itens destacados com * estão disponíveis apenas ao WinCC.

Nesse projeto, a escolha dos softwares WinCC e ScadaBR foi realizada porque o primeiro já é utilizado há anos pela Cagepa, enquanto que o segundo possui o atrativo de ser software livre. No ScadaBR foi feita uma tela para o conjunto motobomba do Sierra que foi replicada em funcionalidades no WinCC. No WinCC, a tela do sistema precisa ser compilada para as alterações serem aplicadas. No ScadaBR as alterações precisam apenas ser salvas e ficam disponíveis de modo imediato independente de haver ou não erros. Portanto, no ScadaBR é necessário verificação de todos os itens programados afim de garantir sua funcionalidade.

### 3.1.1 Tag

No contexto de sistemas SCADA as *Tags* são objetos que contêm valores e estão associadas a uma interface de comunicação. Podemos nesse ponto definir dois novos conceitos que são utilizados em qualquer sistema SCADA. O primeiro deles é o de *Datasource* como sendo a origem dos dados. Um *Datasource* compreende uma conexão feita entre equipamento de campo e o sistema SCADA através de protocolo comum a ambas extremidades. O segundo conceito é o de *Datapoints* que são as *Tags* de fato. As *Tags* representam valores em tempo real de objetos físicos, como o valor bruto ou tratado (também chamado de valor de engenharia) de um sensor de nível ou a velocidade do motor, por exemplo. No entanto, eles também podem ser usados para representar dados provenientes de outras fontes, como um banco de dados SQL.

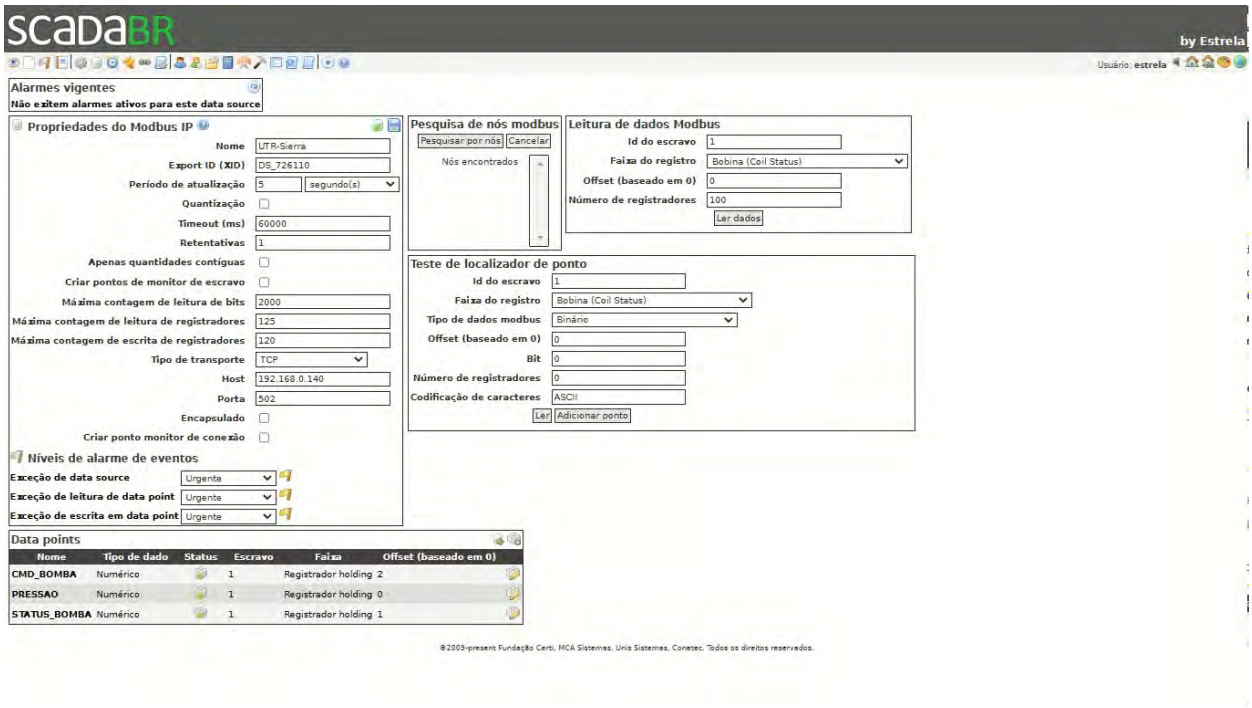
A criação das *Tags* desse projeto pode ser feita diretamente entre o servidor Arduino e o ScadaBR porque ambos possuem protocolo Modbus IP. Na figura 3.3 podemos ver as configurações necessárias e a ilustração dos conceitos de *Datasouces* e *Datapoints*.

No WinCC as *Tags* para o servidor Arduino foram feitas indiretamente porque o WinCC não dispõem do protocolo Modbus IP. Uma alternativa para estabelecer a comunicação é utilizar um CLP Siemens como *bridge* entre servidor Arduino e o WinCC, conforme a figura 3.4.

### 3.1.2 Telas

Esta etapa trata do desenvolvimento das telas de ambos os sistemas SCADA. Os objetos, ou componentes, gráficos que são inseridos nas telas podem ser do tipo estático ou dinâmico. Um objeto

**Figura 3.3:** *ScadaBR Datasource e Datapoints*




*Fonte: Autor*

**Figura 3.4:** *Bridge servidor Arduino e WinCC*

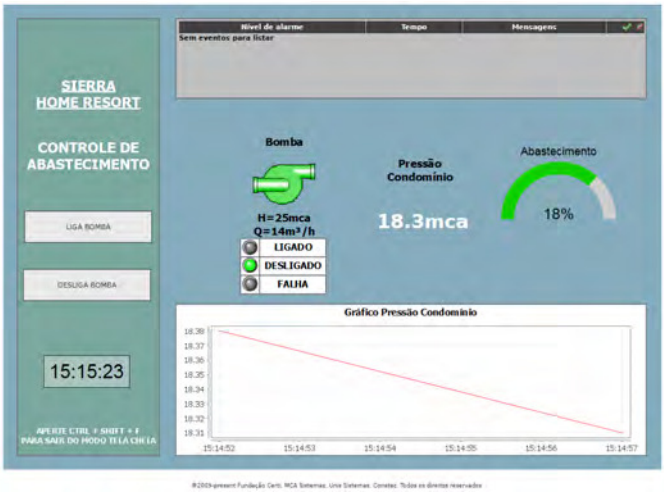

*Fonte: Autor*

estático não possue animação e está na tela geralmente para criar o que chamamos de *background* ou imagem de fundo. Um objeto dinâmico, por sua vez, possue propriedades (cor, texto, visibilidade etc.) conectadas ao valor da *Tag* que esteja nele associada. Essa conexão pode ser de várias maneiras. Em geral, a *Tag* pode animar uma objeto de maneira *input*, *output* e *input/output*.

- *Input:* O objeto é capaz de escrever na *Tag*.
- *Output:* O objeto recebe o valor contido na *Tag*.
- *Input/Output:* O objeto recebe o valor contido na *Tag* e é capaz de escrever na Tag.

A seguir são apresentadas as telas que são exibidas no WinCC e ScadaBR para prover o controle do conjunto motobomba do Sierra. No ScadaBR, figura 3.5, a interação das *Tags* nos objetos é feita por meio de *Scripts* e *Eventos* que alteram a forma ou visualização do objeto.

**Figura 3.5:** *ScadaBR sinóptico para conjunto motobomba*




*Fonte: Autor*

O WinCC, figura 3.6 usa propriedades *build-in* dos próprios objetos e fica transparente ao programador a forma de animação do objeto. A qualidade gráfica dos objetos no WinCC são mais robustas e aparentam um melhor conforto visual.

**Figura 3.6:** *WinCC sinóptico para conjunto motobomba*


*Fonte: Autor*

## 3.2 Hardware de campo

Após a fase de desenvolvimento de software foi iniciado os trabalhos de montagem do hardware do servidor Arduino no condomínio. O servidor Arduino foi instalado dentro de uma caixa hermética de plástico de 26x24x9cm apropriada para ficar ao tempo. Na figura 3.7 podemos ver a montagem mecânica do servidor ao lado do quadro de acionamento do conjunto motobomba.

**Figura 3.7:** *Montagem do servidor Arduino*

*Fonte: Autor*

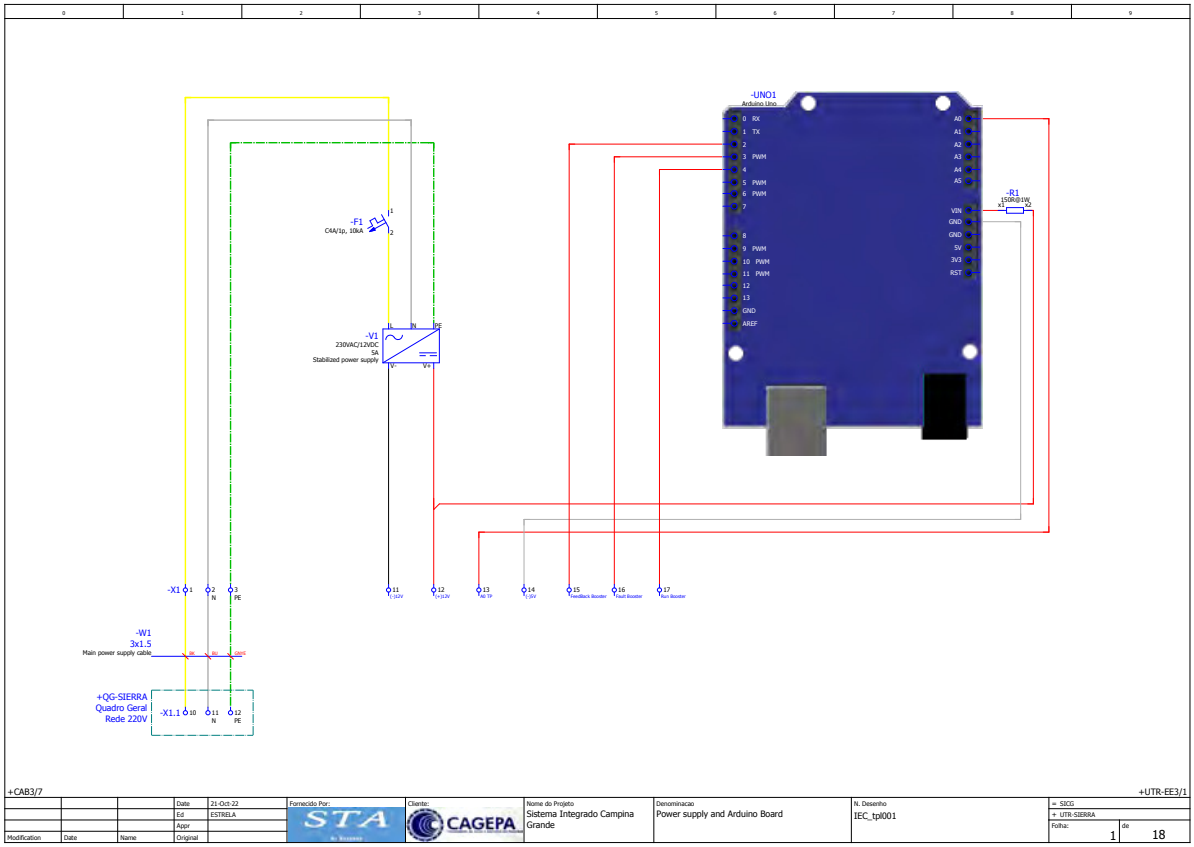
O passo seguinte foi a montagem elétrica da respectiva caixa. A fim de rastrear as principais conexões e documentar o trabalho, o respectivo diagrama multifilar do servidor foi feito.

## 3.3 Infraestura de comunicação

A infraestrutura de comunicação compreende a rede necessária para estabelecer comunicação entre o servidor Arduino e os respectivos sistemas SCADA's. A escolha da tecnologia de transmissão de dados deve ser feita com base no custo. VPNs permite utilizar links de provedores de internet mas possuem a desvantagem do custo fixo permanente. WLAN são redes proprietárias que apresentam alto custo inicial de investimento. Esta segunda opção foi preferível porque a Cagepa já possui a infraestrutura de rede sem fio em grande parte da cidade.

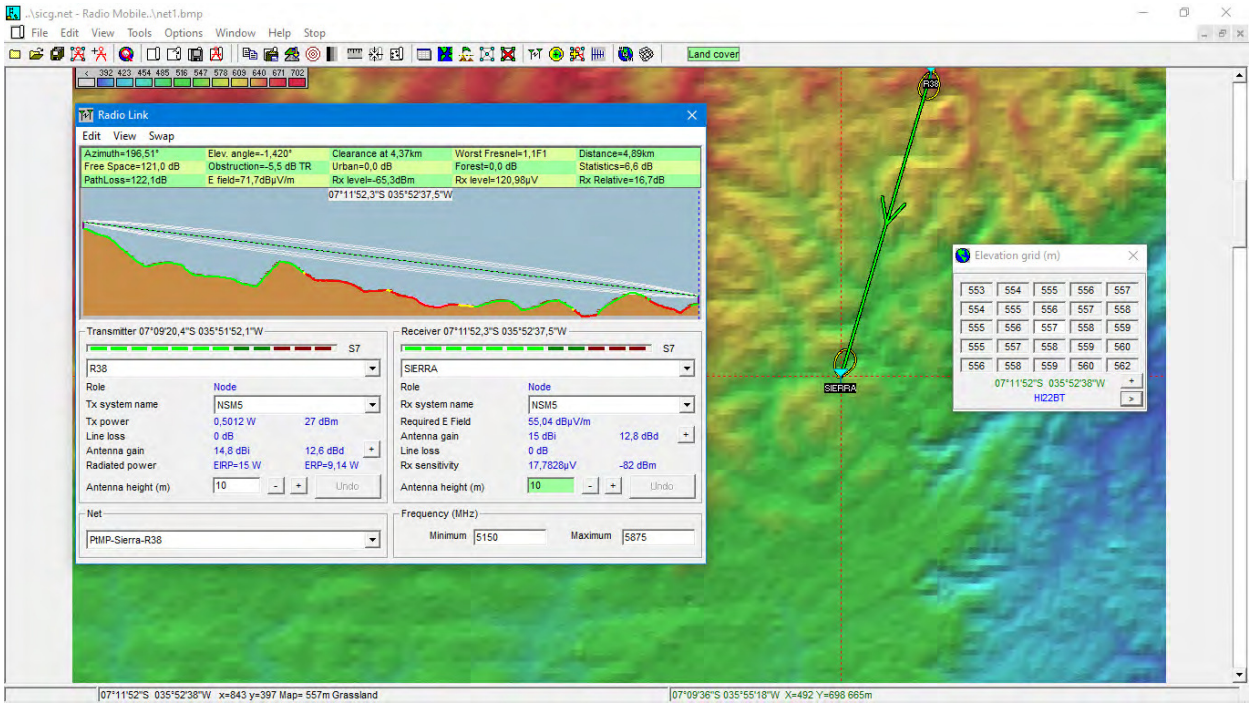
O cálculo de rádio enlace necessária para estabelecer a comunicação foi feita utilizando os conceitos da disciplina Comunicações sem fio e o software gratuito rádio mobile. O rádio mobile possui várias configurações avançadas que permitem obter a topografia entre os links, cadastrar equipamentos e topologia de redes, perfis de irradiação das antenas e resumo final de viabilidade da comunicação.

**Figura 3.8:** *Diagrama elétrico*




*Fonte: Autor*

**Figura 3.9:** *Cálculo do rádio enlance*




*Fonte: Autor*

# Capítulo 4

# Conclusão

O desenvolvimento deste trabalho objetivou resolver um problema real de gestão operacional de abastecimento hidráulico de um condomínio utilizando um software profissional e amplamente usado por algumas indústrias. O ScadaBR foi usado para o desenvolvimento de uma HMI que possibilitasse controlar um sistema de bombeamento através de um servidor Arduino. Com o desenvolvimento contínuo do software ScadaBR, é fornecido ao programador uma série de novas funcionalidades e atuação de forma mais rápida sobre a planta. Vários conceitos e técnicas de programação necessitou ser absorvido pelo aluno para execução desse trabalho. Em particular, a respeito do servidor Arduino.

As possíveis funcionalidades de exportar dados, imprimir relatórios e histórico nos gráficos apresentados em tela são um diferencial do ScadaBR mostrando a alta aplicabilidade do sistema na indústria, para um melhor desempenho do sistema e estudo de caso para avanços nos processos produtivos.

Para aplicações acadêmicas o tal software pode ser adotado para facilitar estudos de controle de malha, obtenção de tabelas para funções de transferências dos sistemas além da aplicabilidade do estudo do software em disciplinas que adotam o desenvolvimento de HMI's em seus laboratórios.

Para trabalhos futuros é proposto uma melhoria no *hardawe* e *software* do servidor Arduino. Uma mehoria possível no *hardware* é compactar o fornecimento de energia num único dispositivo. O uso de *ethernet shield* com o padrão IEEE 802.3af permitiria alimentar todo o servidor Arduino pela interface ethernet e, assim, eliminaria pelo menos um item do hardware, diminuindo o custo. No software é sugerido uma possível implementação para executar duas instâncias simultâneas do Modbus, isto é, "rodar"no mesmo código um *socket* servidor e outro *socket* cliente. Isso possibita o Arduino responder a requisições de clientes e, ao mesmo tempo, ser a origem de solicitações para outro sistema.

# Apêndice A

# Base de Dados

Capítulo opcional usado para que o autor coloque um texto,documento ou informação referente ao assunto do trabalho, mas que não deve interromper a leitura.

# Referências Bibliográficas

[GOMES 2009] GOMES, H. P. Sistemas de bombeamento. *João Pessoa, Editora Universitária UFPB*, 2009. 3

[Silveira e Lima 2003] SILVEIRA, L.; LIMA, W. Q. Um breve histórico conceitual da automação industrial e redes para automação industrial. *Redes para Automação Industrial. Universidade Federal do Rio Grande do Norte*, p. 16, 2003. 1

[Swales 1999] SWALES, A. Open modbus/tcp specification, release 1.0. *Schneider Electric*, 1999. 10, 11

[Vianna, BRINGHENTI e MARTINS 2008] VIANNA, W. d. S.; BRINGHENTI, P.; MARTINS, L. Sistema scada supervisório. *Instituto Federal Fluminense de Educação Ciência e Tecnologia. Campos dos Goytacazes, Rio de Janeiro*, 2008. 3

**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA**
Campus Campina Grande
R. Tranqüilino Coelho Lemos, 671, Dinamérica, CEP 58432-300, Campina Grande (PB)
CNPJ: 10.783.898/0003-37 - Telefone: (83) 2102.6200

# Documento Digitalizado Ostensivo (Público)

## Entrega da versão final do TCC

| | |
|---|---|
| **Assunto:** | Entrega da versão final do TCC |
| **Assinado por:** | Luciano Lacerda |
| **Tipo do Documento:** | Dissertação |
| **Situação:** | Finalizado |
| **Nível de Acesso:** | Ostensivo (Público) |
| **Tipo do Conferência:** | Cópia Simples |

Documento assinado eletronicamente por:

- **Luciano Estrela de Lacerda**, **ALUNO (202011210025) DE TECNOLOGIA EM TELEMÁTICA - CAMPINA GRANDE**, em 20/02/2024 09:52:15.

Este documento foi armazenado no SUAP em 20/02/2024. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1086927
Código de Autenticação: 0fcf291276